UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

UNIVERSITAT POLITÈCNICA DE CATALUNYA

ESEIAAT

Desenvolupament d'algorismes de planificació de camins amb Robotino i ROS

Juny de 2018

Presenta:
GUILLERMO OSIO SILVA

Tutor:
ALBERT MASIP ALVAREZ

Agraïments

Amb especial dedicatòria a la meva família i als meus amics.

La finalització d'aquest treball significa el final d'una etapa i el principi d'una altre.

Vull donar les gràcies a aquelles persones que han estat al meu costat aquests anys.

També m'agradaria agrair l'ajuda donanda, durant l'elaboració d'aquest treball, per Albert Masip i Albert Márquez.

Índex

Índex de figures

Capítol 1

Introducció

Aquest Treball de Final de Grau (en endavant TFG) és el resultat del procés aplicat al desenvolupament d'algoritmes per la planificació de camins amb *Robotino* i *Robot Operating System (ROS)*. La finalitat principal és donar a conèixer la planificació de camins i supervisió mitjançant un robot mòbil.

Actualment a la universitat s'utilitza en algunes assignatures el *Robotino* amb el sistema operatiu de *Windows*. La aportació d'aquest treball ha sigut migrar a un software més eficient com és *Linux* amb *Robot Operating System*.

Avui en dia, la tècnica de planificació de camins ha cobrat rellevància en el sector industrial, de tal manera que l'automatització és imprescindible.

La raó que ha motivat l'elaboració d'aquest TFG ha estat relacionat amb la idea de millorar l'ús del *Robotino* mitjançant *Robot Operating System*. A més a més, he tingut l'oportunitat d'apendre tècniques per donar visió a un robot mòbil perquè tingués una conducció autònoma.

Aquest TFG s'ha organitzat de la següent manera.

- Per iniciar es presenta la introducció, els objectius, el context i el pla de treball.
- A continuació es presenta l'arquitectura de treball.
- Posteriorment, n'hi ha tres capítols amb el contigut de la recerca.
- Finalment, es poden trobar les conclusions, la bibliografía i els annexos que presenten totes les funcions utilitzades.

1.1 Objectius

Els principals objectius d'aquesta recerca són:

- Aplicar algoritmes de planificació de camins per un robot mòbil.
- Utilitzar un sistema operatiu que ha cobrat rellevància com és *Robot Operating System*.
- Potenciar els coneixements de programació amb *MATLAB* amb la finalitat de controlar un robot mòbil.
- Crear una eina visual per poder supervisar un procés en temps real.
- Aconseguir que un robot mòbil faci el seguiment d'una trajèctoria.

1.2 Context

Actualment l'automatització és una pràctica que s'està estenent en totes direccions. Les grans empreses estàn donant importància als vehicles autònoms. A més a més, en les indústries s'estan substituint els treballadors per robots mòbils per a tasques de transport.

S'ha disposat de diferents documents de les assignatures, *Control i guiatge de robots mòbils* i *Planificació, simulació i supervisió de processos*, que han aportat una part important de la informació teòrica.

L'espai de treball ha sigut el *laboratori d'informàtica industrial* ubicat al TR11 del campus universitari de *Terrassa*.

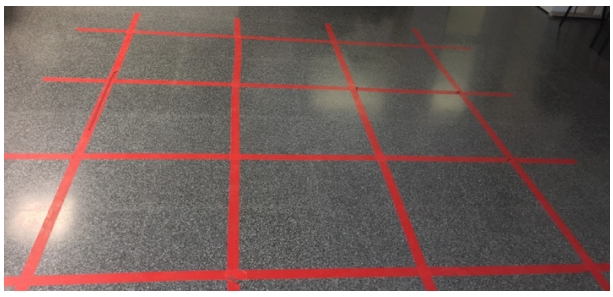

Figura 1.2.1: Espai de treball.

1.3 Pla de treball

En primer lloc, es va realitzar un pla de treball que es va modificar a causa d'algunes dificultats.

La tasca de posar en marxa la plataforma per treballar es va allargar degut a problemes amb la instal·lació.

Les tasques relacionades amb l'assignatura de *Control i guiatge de robots mòbils*, esmentades en aquest pla, va ser un aprenentatge autònom ja que no han format part dels estudis cursats.

Les tasques relacionades amb l'assignatura de *Planificació, simulació i supervisió de processos* es van realitzar de forma simulatànea, aprofitant algunes activitats.

Finalment, les diferents tasques s'han organitzat, des del principi del quadrimestre, de la següent manera:

Setmana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Posta en marxa de la plataforma																
Processat de la Imatge																
Seguiment de línia + Gir																
Planificació de camins																
Interfície gràfica d'usuari																

Figura 1.3.1: Pla de treball.

Capítol 2

Arquitectura

2.1 Descripció de l'entorn de treball

Aquest projecte de recerca s'ha realitzat a partir del software *Robot Operating System* sobre un sistema operatiu *Ubuntu 16.04 LTS*. Per desenvolupar les diferents tasques planificades, s'ha utilitzat l'eina de programació de *MATLAB* amb l'objectiu de controlar un robot mòbil. El robot s'anomena *Robotino* fabricat per *Festo Didactic*.

2.2 Robotino

Robotino és un robot mòbil fabricat per *Festo Didactic* amb finalitats acadèmiques i d'investigació [13]. *Robotino* es basa en una conducció omnidireccional, això permet que el sistema es mogui de forma lliure. Aquest robot incorpora un ordinador a bord amb sistema operatiu Linux, el qual li permet un control robust per realitzar una conducció autònoma. A través d'un xarxa WLAN-Link, *Robotino* pot enviar totes les lectures dels sensors a un PC extern. De forma inversa, des de un PC extern, les comandes de control poden ser emeses.

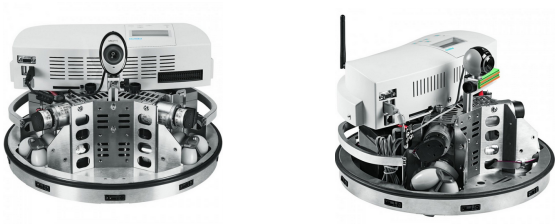

Figura 2.2.1: *Robotino*.

A nivell de Hardware, *Robotino* és un dispositiu molt ben equipat. La seva conducció omnidireccional consta de tres rodes *Mecanum* controlades de forma individual i separades 120°. Disposa d'un sensor de para-xocs. A més a més, incorpora diferents sensors de proximitat del fabricant *Sharp*. Distribuits al voltat de la seva circumferència. També incorpora una càmera de resolució QVGA com a sistema de visió.

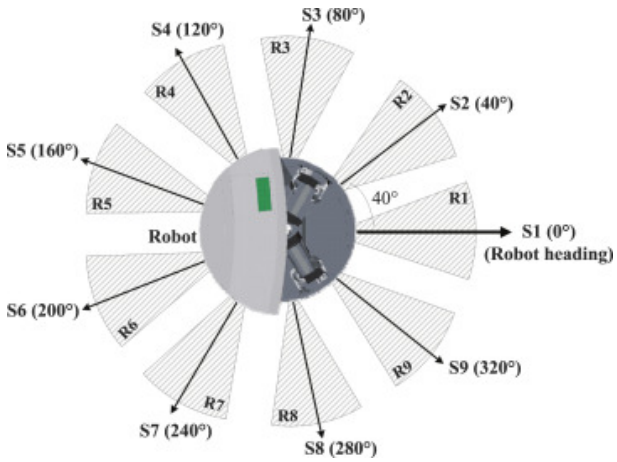

Figura 2.2.2: Distribució dels sensors de proximitat en el *Robotino*.

2.3 Robot Operating System

2.3.1 Definició

Robot Operating System, més conegut com *ROS*, és una estructura de programari de codi obert que permet el desenvolupament de programes per a robots, proveint funcionalitat semblant a un sistema operatiu.
ROS implementa els serveis típics d'un sistema operatiu tals com abstacció de hardware, control de dispositius de baix nivell, implementació de les funcionalitats més comuns, intercanvi de missatges entre processos i gestió de paquets.
ROS és un sistema operatiu que s'executa sobre un altre sistema operatiu, en aquest cas, *Ubuntu* 16.04 *LTS*.

2.3.2 Funcionament

ROS esta format per un número de nodes independents on cadascún es comunica amb la resta utilitzant el model de publicador-subscriptor.
Els publicadors envien informació al dispositiu, per exemple, excitar els motors, mitjançant **rospublisher**.
Els subscriptors poden rebre informació del dispositiu, per exemple, rebre una imatge, mitjançant **rossubscriber**.
Els nodes es poden comunicar entre ells utilitzant **rosmessage**.

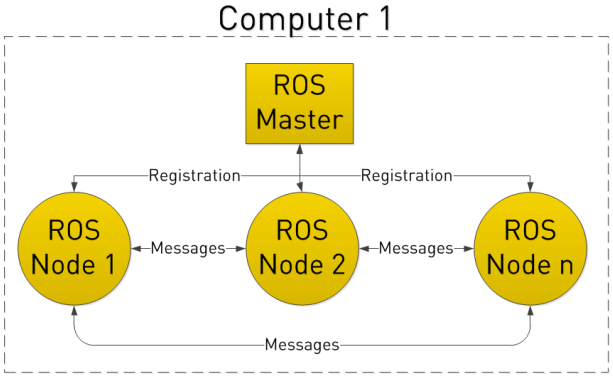

Figura 2.3.1: Funcionament dels nodes a *ROS*.

2.3.3 Instal·lació

El procés d'instal·lació per poder treballar amb el *Robotino* mitjançant *ROS* [18] [19] es detalla a l'apartat d'Annexes. Es recomana treballar amb un equip en condicions inicials o màquina virtual. Cal afegir que es un procediment complex que consta de les següents etapes:

1. Instal·lació del sistema operatiu *Ubuntu* 16.04 *LTS* [14].
2. Instal·lació del sistema operatiu *ROS Kinetic* [15].
3. Creació de l'espai de treball per a ROS, *catkin_ws* [16][17].
4. Creació de *CFcard* per poder utilitzar el *Robotino* [20].
5. Configuració del *Robotino* per poder tenir accés a la xarxa.
6. Instal·lació de API2 dintre del *Robotino* [21].
7. Instal·lació de API2 a l'equip [21][22][23].

8. Instal·lació del paquet de drivers de *ROS* per *Robotino* a l'equip.

9. Finalització de la instal·lació.

Una vegada el procés d'instal·lació hagi finalitzat sense errors, es pot procedir a arrencar el sistema dins d'un terminal [24].

$$roscore$$
$$roslaunch\ robotino\_node\ robotino\_node.launch\ hostname := IP\_ROBOTINO$$

A més a més podem utilitzar altres serveis com per exemple visualitzar la càmera en temps real o controlar el robot mitjançant el teclat.

$$roslaunch\ robotino\_teleop\ keyboard\_teleop.launch$$
$$rosrun\ image\_view\ image\_view\ image := image\_raw$$

2.4 MATLAB

MATLAB és una eina de software matemàtic que ofereix un entorn de computació numèrica i un llenguatge de programació [25]. La *toolbox*, "*Robotics System Toolbox*"[26], permet vincular *ROS* amb *MATLAB*.

Existeixen diferents comandes per utilitzar els diferents serveis de *ROS* dins d'aquest software:
Per iniciar *ROS* es necessari invocar la comanda **rosinit** a la *Comand Window*.

Per visualitzar els nodes disponibles, **rosnode list**.

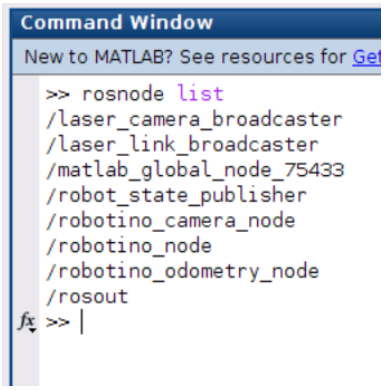

Figura 2.4.1: Nodes disponibles.

Dins dels nodes disponibles, existeixen diferents "*topics*"per utilitzar, **rostopic list**.

Figura 2.4.2: "*topics*"disponibles.

Per visualitzar les característiques de cada "*topics*", **rostopic info /topic.**

Per utilitzar aquests "topicsçal utilitzar el missatge apropiat. Existeixen una gran quantitat, es poden visualitzar amb **rosmsg list**.

Els diferents nodes poden ofereir una varietat de servei, per visualitzar-los cal utilitzar **rosservice list**.

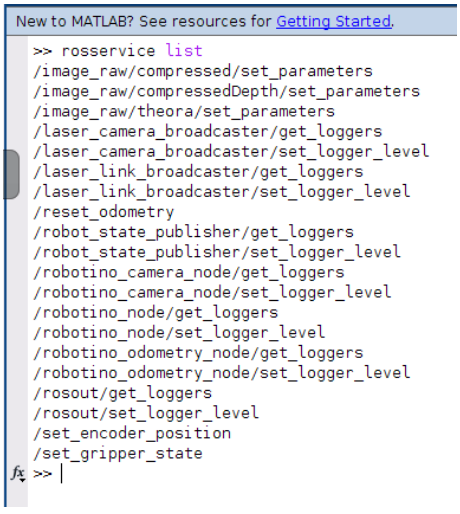

Figura 2.4.3: Serveis disponibles.

Per aturar aquest servei es necessari utilitzar **rosshutdown**.

Capítol 3

Visió

3.1 Introducció

Un dels objectius d'aquest treball és aconseguir guiar el robot a través dels diferents camins que pot formar una graella. El Robotino ha de ser capaç de fer el seguiment perfecte d'una línia situada al terra.

El robot disposa d'una càmera, situada a la part frontal, que captura imatges a color amb una resolució de QVGA, és a dir, de 320x240 píxels. Aquesta càmera ha d'estar orientada al terra.

Per poder fer el seguiment de la línia s'ha de resoldre el problema de conèixer l'angle i la distància que separa el robot de la línia a través d'una imatge capturada.

3.2 Transformacions i rotacions d'eixos. Calibració de la càmera

En primer lloc, és important conèixer quines transformacions (rotacions i translacions) s'han de realitzar per tal de passar d'un punt expressat en coordenades del sistema de referència del robot a expressar-lo en coordenades de la imatge. A continuació, s'ha de resoldre el problema invers, saber quines transformacions s'han de realitzar per saber quin punt a l'espai del Robotino representa un píxel de la imatge [1].

Durant el procés de calibració del robot, tenen lloc quatre sistemes diferents de referència.

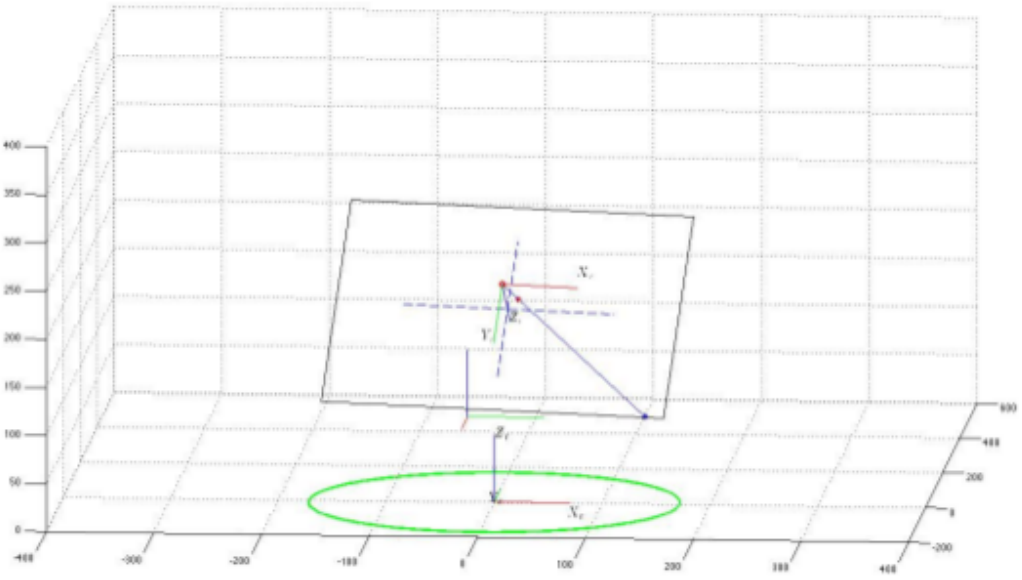

Figura 3.2.1: Representació dels diferents sistemes de refèrencia del Robotino.

1. Sistema de referència propi del robot.
2. Sistema de referència de calibració, desplaçat i rotat $-90°$.
3. Sistema de referència de la càmera aixecat en z.
4. Sistema de referència de la imatge 2D.

Aquest procés té la finalitat de conèixer les relacions existents entre aquests sistemes de refèrencia per tal de saber la ubicació en tot moment del Robotino. *Vision Caltech* disposa d'una eina de la calibració, *Camera Calibration Toolbox* [12], que s'ha fet servir per conèixer els paràmetres propis de la càmera del robot.

3.3 Paràmetres interns de la càmera

3.3.1 Paràmetres Intrínsecs de la càmera

Els paràmetres intrínsecs són aquells que defineixen la geometría interna i la òptica de la càmera. Sempre són constants mentre no es produexi un canvi en la orientació del seu centre òptic. Aquests paràmetres venen expressats en la matriu de paràmetres intrínsecs de la càmera que rep el nom de KK,

$$KK = \begin{bmatrix} fc(1) & \alpha_c * fc(1) & cc(1) \\ 0 & fc(2) & cc(2) \\ 0 & 0 & 1 \end{bmatrix}$$

on,

- fc és la longitud o distància focal, expressada en píxels i emmagatzemada en un vector de dues components x,y.
- cc són les cordenadees x,y del centre òptic o punt principal.
- α_c és el coeficient d'inclinació que correspon a l'angle format entre els eixos x i y dels píxels.

Per obtenir aquests paràmetres és necessari fer ús d'un escaquer i capturar imatges amb la càmera del Robotino en diferents posicions. És important utilitzar la funció de *MATLAB imwrite/imread* [27][28] per evitar veure els eixos generats de forma automàtica a l'hora de guardar una imatge de forma directa.

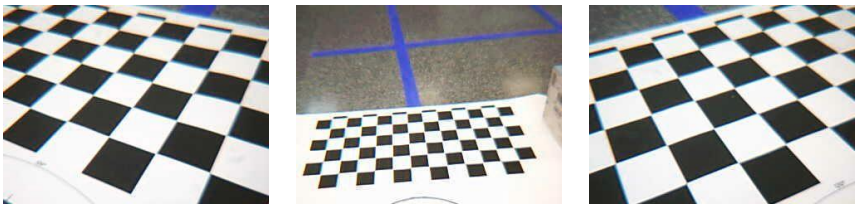

Figura 3.3.1: Imatges utilitzades per a la calibració intrínseca.

Ara doncs, s'ha de executar l'eina de calibració de *Caltech*, *calibgui* seleccionar l'opció *Calibration* i seguir amb els passos de l'assistent. Una vegada ha finalitzat el procés, s'han obtingut els diferents paràmetres interns de la càmera.

$$KK = \begin{bmatrix} 438,4386 & 0 & 125,7732 \\ 0 & 253,15647 & -54,1278 \\ 0 & 0 & 1 \end{bmatrix}$$

3.3.2 Paràmetres Extrínsecs de la càmera

Els paràmetres extrínsecs de la càmera són aquells que relacionen els sistemes de referència del món real i de la càmera, descrivint la posició i orientació de la càmera en el sistema de referència de coordenades del món real.

Per obtenir aquestes dades, igual que en l'apartat anterior, es treballa amb una imatge de l'escaquer però en aquest cas en una posició coneguda.

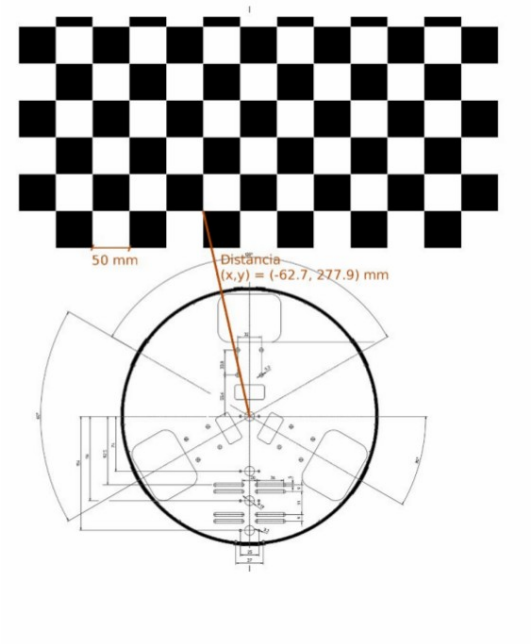

Figura 3.3.2: Posició recomenada per capturar la imatge.

S'ha de procedir a executar l'eina de *Caltech*, *extrinsic_computation* i seguir els passos de l'assistent.

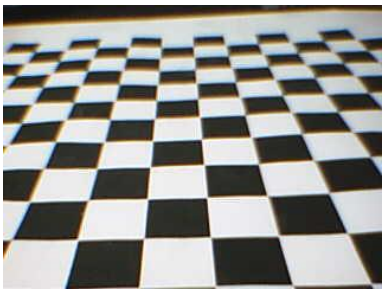

Figura 3.3.3: Imatge utilitzada per a la calibració extrínseca.

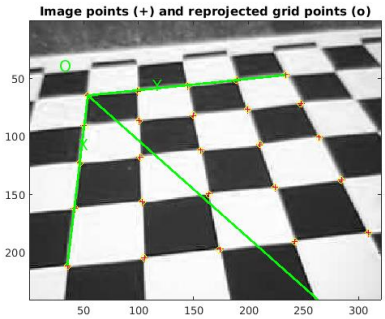

Figura 3.3.4: Imatge resultant de la calibració extrínseca.

El resultat d'aquesta operació són els paràmetres extrínsecs de la calibració.

$$Tc\_Ext = \begin{bmatrix} -79,36659 \\ 219,75919 \\ 467,77044 \end{bmatrix}$$

$$Rc\_Ext = \begin{bmatrix} 0,089789 & 0,98230038 & -0,1643891 \\ 0,5880279 & -0,1855026 & -0,7872813 \\ -0,80384138 & -0,025975749 & -0,594276 \end{bmatrix}$$

on

- Tc_Ext és el vector de Translació.
- Rc_Ext és la matriu de Rotació.

Segons els programadors d'aquesta eina de calibratge per expressar un punt en coordenades del sistema de calibració en coordenades de la càmera, s'ha d'interpretar de la següent manera:

$$Pcam = Rc\_ext * Pcal + Tc\_ext$$

3.4 Pas de coordenades robot a coordenades de la imatge

Tal i com s'ha explicat al segon apartat d'aquest capítol, existeix la necessitat de saber a quin punt de la imatge equival un punt del robot. Per tal de resoldre aquest problema, s'han d'aplicar diverses transformacions de coordenades en funció dels diferents sistemes de referència.

3.4.1 Pas de coordenades robot a coordenades de calibració

Per efectuar el pas de coordenades robot a coordenades de calibració s'ha de fer una translació d'eixos i una rotació respecte l'eix z.

En coordenades homogènies, una translació d'un punt es realitza mitjançant una matriu de translació amb la forma:

$$T_v = \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Per efectuar la rotació en un angle θ d'un punt en coordenades homogènies sobre el pla cal utilitzar la matriu:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Tal i com es veu a les figures 3.3.4 que és el resultat de la calibració extrínseca i 3.3.2 on es mostra un punt conegut de l'escaquer respecte el centre del robot, s'arriba a la conclusió de que cal efectuar una translació de $[-62.7, 477.9, 0]^T$ i una rotació de 90°. Com que el que es transformen són els eixos, caldrà prendre la translació canviada de signe i l'angle de rotació de −90°.

$$T_{rob \to cal} = \begin{bmatrix} 1 & 0 & 0 & -(-62.7) \\ 0 & 1 & 0 & -(477.9) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; R_{rob \to cal} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.4.2 Pas de coordenades del robot a coordenades de la càmera

Per efectuar el pas de coordenades robot a coordenades de la càmera s'ha de realitzar la següent operació,

$$P_{cam} = T_{cal \to cam} * R_{cal \to cam} * R_{rob \to cal} * T_{rob \to cal} * P_{rob}$$

on,

- P_{cam} és un punt 3D expressat en coordenades homogènies en el sistema de referència de la càmera.
- P_{rob} és un punt 3D expressat en coordenades homogènies en el sistema de referència del robot.
- $T_{cal \to cam}$ és la matriu de translació Tc_ext, proporcionada a l'hora de fer la calibració extrínseca, en coordenades homogènies.
- $R_{cal \to cam}$ és la matriu de rotació Rc_ext, proporcionada a l'hora de fer la calibració extrínseca, en coordenades homogènies.
- $T_{rob \to cal}$ és la matriu de translació definida en l'apartat anterior.
- $R_{rob \to cal}$ és la matriu de rotació definida en l'apartat anterior.

$$T_{rob \to cal} = \begin{bmatrix} 1 & 0 & 0 & -(-62.7) \\ 0 & 1 & 0 & -(477.9) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{rob \to cal} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{cal \to cam} = \begin{bmatrix} 1 & 0 & 0 & -79,3665 \\ 0 & 1 & 0 & 219,7591 \\ 0 & 0 & 1 & 467,7704 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{cal\to cam} = \begin{bmatrix} 0,089789 & 0,982300 & -0,164389 & 0 \\ 0,588027 & -0,185502 & -0,787281 & 0 \\ -0,803841 & -0,025975 & -0,594276 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.4.3 Pas de coordenades robot a coordenades de la imatge

Per efectuar el pas de coordenades robot a coordenades de la imatge s'ha de realitzar la següent operació:

$$\lambda * P_{imatge} = [KK, [0,0,0]^T] * T_{cal\to cam} * R_{cal\to cam} * R_{rob\to cal} * T_{rob\to cal} * P_{rob}$$

Si evaluem aquesta equació al voltant del punt $P_{rob} = [-62.7, 277.9, 0, 1]^T$, el resultat obtingut és $\lambda * P_{imatge} = [11688.733, 68788.722, 307]^T$.
Com P_{imatge} és un punt 2D expressat en coordenades homogènies, cal que la seva tercera component valgui 1, és a dir, el valor de $\lambda = 307$. Aquest paràmetre té relació amb la profunditat i la distància focal.

Si realitzem, $P_{imatge} = \lambda * P_{imatge}/\lambda$, obtenim que $P_{imatge} = [38.07, 224.067, 1]^T$, on podem comprobar com fan referència al mateix punt en sistemes de refèrencia diferents.

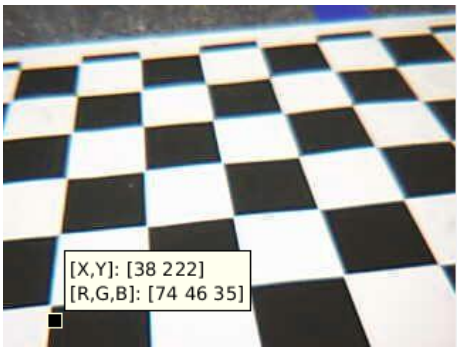

Figura 3.4.1: Imatge de comparació del mateix punt en sistemes de coordenades diferents.

3.5 Pas de coordenades imatge a coordenades del robot

Existeix la necessitat de conèixer quin punt en coordenades Robotino correspon un determinat píxel de la imatge.

$$M = T_{cal \to cam} * R_{cal \to cam} * R_{rob \to cal} * T_{rob \to cal}$$

L'equació de l'apartat anterior per determinar el pas de coordenades robot a coordenades de la imatge es pot reescriure de tal manera que,

$$\lambda * P_{imatge} = [KK, [0,0,0]^T] * M * P_{rob}$$

Sí el nostre problema és saber quin punt en coordenades Robotino correspon un determinat píxel de la imatge, caldra doncs manipular aquesta equació.

$$P_{rob} = [KK]^{-1} * M^{-1} * \lambda * P_{imatge}$$

Es considera de forma pràctica la matriu M de tal manera que,

$$M = \begin{bmatrix} R_{3x3} & T_{3x3} \\ 0_{1x3} & 1 \end{bmatrix}$$

on,

- R és la part de la matriu M corresponent a la rotació.
- T és la part de la matriu M corresponent a la translació.

Amb aquesta condició és pot tornar a reescriure la equació com

$$\lambda * P_{imatge} = KK * R * P_{rob} + KK * T$$

Obtenim que

$$P_{rob} = R^T * KK^{-1} * \lambda * P_{imatge} - R^T * T$$

El valor de la profunditat λ és desconegut, caldrà prendre la tercera componenent de la equació anterior i igualar-la a zero ja que el robot es mou únicament en el pla x-y.

$$[R^T * KK^{-1} * P_{imatge}]_z * \lambda - [R^T * T]_z = 0$$

d'on es pot obtenir el valor de λ.

$$\lambda = \frac{[R^T * T]_z}{[R^T * KK^{-1} * P_{imatge}]_z} \tag{3.1}$$

Amb el valor de la profunditat conegut, podem aplicar la següent igualtat per conèixer un P_{robot} a partir d'un P_{imatge}.

$$P_{rob} = R^T * KK^{-1} * \lambda * P_{imatge} - R^T * T$$

3.6 Tractament de la imatge

Per poder resoldre el problema de conèixer la distància que separa el robot respecte la línia, la qual ha de fer el seguiment, és imprescindible realitzar el tractament de la imatge capturada amb la finalitat d'obtenir informació per tal de poder realitzar el control de trajectòria sobre el Robotino [2].
És un procediment que consta de diferents etapes. A continuació es realitza la explicació del procés amb un exemple pràctic.

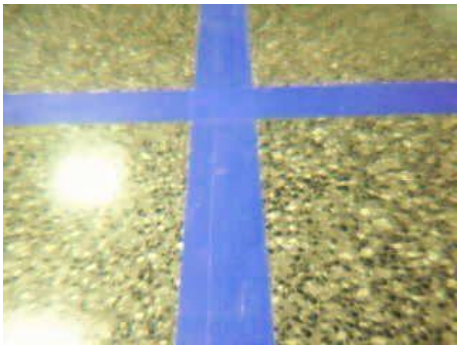

Figura 3.6.1: Exemple de captura per realitzar el tractament de imatge.

El primer tractament a fer és transformar la imatge RGB a l'espai *YCbCr* [29] per aïllar la seva tercera capa. L'espai *YCbCr* és una manera de codificar la informació RGB mitjançant tres capes.

- Component de Luminància *Y*.
- Crominància del color blau *Cb*.
- Crominància del color vermell *Cr*.

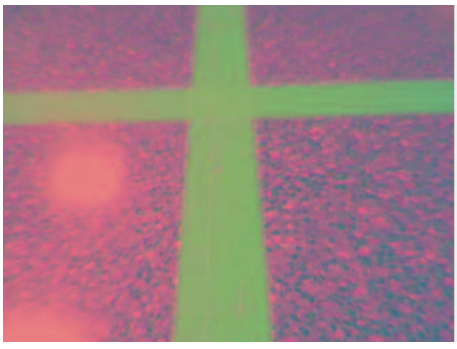

Figura 3.6.2: Capa *Cr* de la figura 3.6.1 en l'espai *YCbCr*.

El següent pas és transformar la capa de crominància del color vermell *Cr* a blanc i negre.

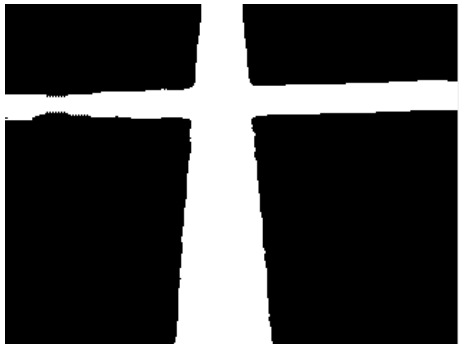

Figura 3.6.3: Figura 3.6.2 transformada a blanc i negre.

Es pot observar com a la imatge inicial, a la part esquerra, s'observa el reflex de la llum a terra i a l'hora de transformar a blanc i negre aquesta pertorbació afecta de tal manera que la línia horitzontal és més estreta.

En aquest punt es procedeix a filtrar per una part la línia vertical i per una altre la horitzontal utilitzant les funcions *strel* [31] i *imerode* [30]. Aquestes funcions permeten erosionar una imatge utilitzant, en aquest cas, formes rectangulars.

Figura 3.6.4: Filtració de la línia vertical respecte la horitzontal.

A més a més, fent ús de la funció *edge* [32], es pot aïllar només el contorn de la línia.

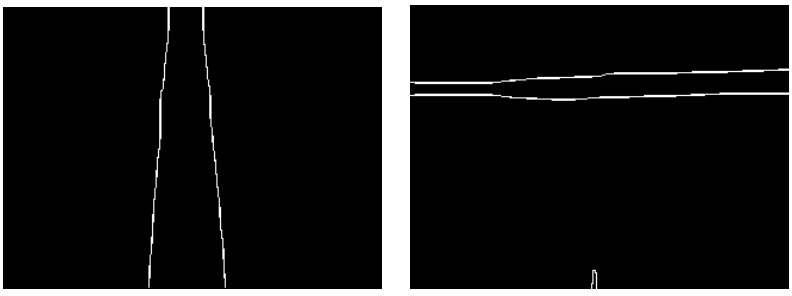

Figura 3.6.5: Filtració del contorn de les línies vertical i horitzontal.

Aquestes línies tenen un gruix de 50 mm aproximadament, el que interessa és que el robot faci el seguiment de la línia just per el centre. L'últim tractament a fer, és obtenir només la línia central respecte el seu contorn. Aquest procediment es realitza aplicant la mitjana aritmètica d'un punt a cada costat del contorn de la línia per obtenir el punt central dins de la matriu de 320x240 píxels que forma la figura 3.6.5.

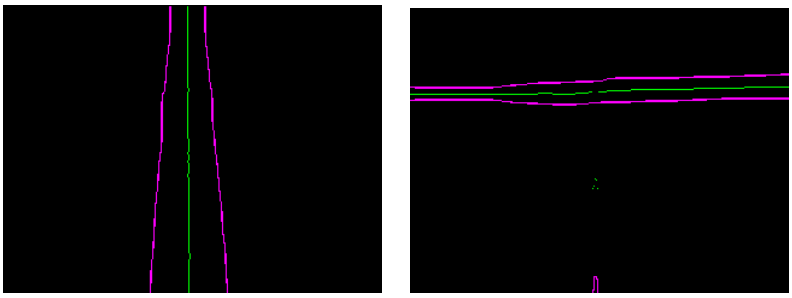

Figura 3.6.6: Filtració de la línia central respecte el seu contorn.

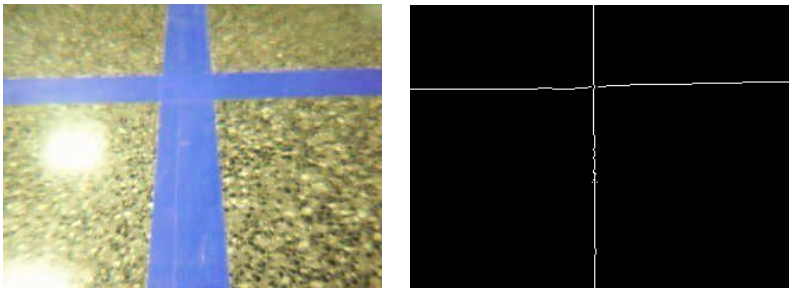

Figura 3.6.7: Comparació de l'inici respecte el final del tractament de la imatge.

En la figura 3.6.7 es pot observar la transformació des de l'inici fins al final del tractament de la imatge.

3.6.1 La transformada de Hough

La transformada de *Hough* és una tècnica per a la detecció de figures en imatges digitals. És una tècnica molt utilitzada dintre del camp de visió per computador. Amb la transformada de *Hough* és possible detectar tot tipus de figures que poden ser expressades matemàticament com per exemple rectes, circumferències o elipses.

En aquest cas s'utilitza la transformada de *Hough* per detectar línies rectes dintre d'una imatge. Aquest algorisme ve definit per la següent expressió.

$$\rho = x * \cos(\theta) + y * \sin(\theta)$$

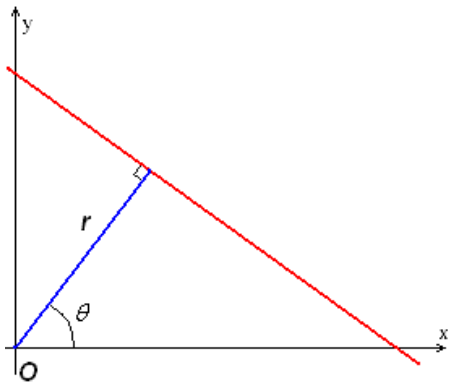

Figura 3.6.8: Representació d'una recta amb la transformada de *Hough*.

Dins de l'espai de *MATLAB* existeixen diferents funcions que utilitzen la transformada de *Hough* per detectar línies. El resultat d'utilitzar les funcions *hough* [33], *houghpeaks* [34] i *houghlines* [35] és un vector que conté la següent informació d'acord amb la definició anterior.

- *Point*1 és el primer punt de la recta trobada en píxels.
- *Point*2 és el segon punt de la recta trobada en píxels.
- *Theta* és l'angle θ que forma la recta respecte l'eix x.
- *Rho* és la longitud ρ de la recta.

En funció de la imatge capturada es pot donar el cas que el tractament inicial no sigui suficientment eficient i a l'hora d'aplicar les funcions de la transformada de *Hough*, aquestes detectin altres linies paràsites.
Per tal de poder seleccionar la recta més adequada s'utilitza un últim filtre. S'ha decidit aplicar la *distància euclidiana* als diferents punts trobats per poder seleccionar el segment més llarg.

La *distància euclidiana* és la distància real que existeix entre dos punts i que es dedueix a partir del teorema de *Pitàgores*.
En un espai bidimensional, la *distància euclidiana* entre els punts P1$(x1, y1)$ i P2$(x2, y2)$ respectivament és

$$d(P1, P2) = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

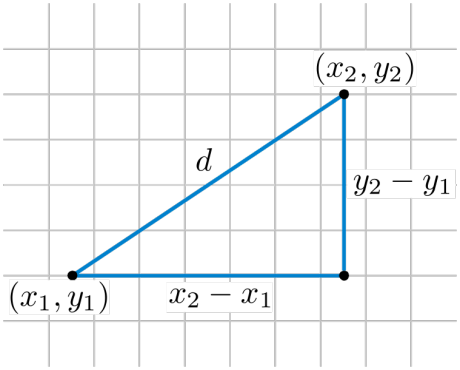

Figura 3.6.9: Distància euclidiana entre dos punts.

Després d'aplicar tot el procediment de tractament de la imatge per aquest exemple, només detecta una línia vertical. Els punts obtinguts per obtenir la recta s'utilitzaran posteriorment per realitzar el càlcul de la distància que separa el robot de la línia.

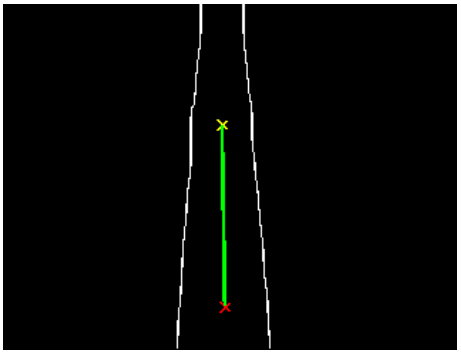

Figura 3.6.10: Recta seleccionada per realitzar el càlcul de la distància.

3.7 Obtenció de l'angle i distància que separen al robot de la línia

A partir de dos punts P1 i P2 en coordenades homogènies i una recta L que els uneix, mitjançant el producte vectorial d'aquests dos punts, realitzant el seu determinant i normalitzant aquests vectors, es pot obtenir la distància d perpendicular que els uneix i l'angle θ que formen.

Si $P1 = [xi, yi]$ i $P2 = [xr, yr]$, el determinant del seu producte vectorial és,

$$L = det \begin{vmatrix} i & j & k \\ xi & yi & 1 \\ xr & yr & 1 \end{vmatrix} = (yi - yr) * i + (xr - xi) * j + (xi * yr - xr * yi) * k$$

Normalitzem L,

$$L = (\frac{yi - yr}{\sqrt{(yi - yr)^2 + (xr - xi)^2}}, \frac{xr - xi}{\sqrt{(yi - yr)^2 + (xr - xi)^2}}, \frac{xi * yr - xr * yi}{\sqrt{(yi - yr)^2 + (xr - xi)^2}})$$

Si es defineix la recta com $L = (\cos\theta, \sin\theta, d)$, obtenim la distància perpendicular d al segment que uneix els punts $P1$ i $P2$,

$$d = \frac{xi * yr - xr * yi}{\sqrt{(yi - yr)^2 + (xr - xi)^2}}$$

i l'angle θ que forma.

$$\theta = \arctan(\frac{\sin\theta}{\cos\theta}) = \arctan(\frac{xr - xi}{yi - yr})$$

Com a resultat final d'aquest capítol, el que s'obtè és una funció que a partir d'una imatge capturada retorna la distància $d1$ que separa el robot respecte la línia vertical, la distància $d2$ que separa el robot respecte la línia hortizontal i l'angle θ que forma respecte la línia vertical.

$$[d1, d2, \theta] = Activitat53(Imatge)$$

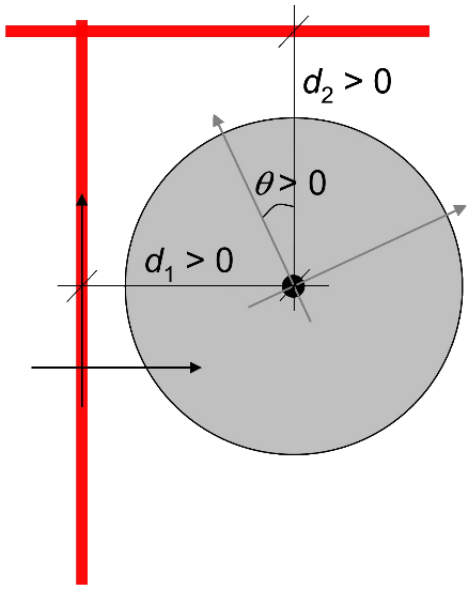

Figura 3.7.1: Criteri d'obtenció dels valors d1, d2 i θ.

Capítol 4

Control i Guiatge

4.1 Control de la trajectòria

Per realitzar el seguiment de la trajectòria, es proposa el mètode de control basat en l'article *"Stability Analysis of Mobile Robot Path Tracking"* [7]. Aquesta estratègia de control està basada en un algoritme que determina geomètricament la curvatura que conduirà el vehicle des d'un punt inicial fins a un punt destí. De tal manera que només varia la seva velocitat angular per modificar la trajectòria, mentre avança a velocitat constant [3][4].

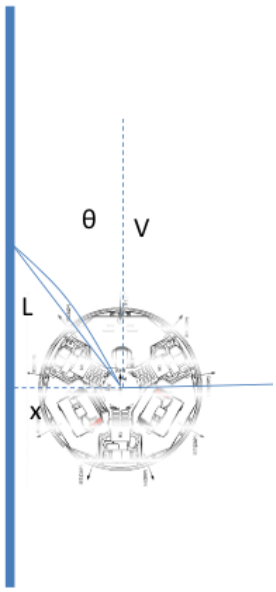

Figura 4.1.1: Control de la trajèctoria.

Aquest mètode de control proposa l'algoritme *"Pure pursuit path tracking"*,

$$\gamma_R = \frac{2x}{L^2} = \frac{2}{L^2}[x\cos(\theta) - \sqrt{L^2 - x^2}\sin(\theta)]$$

on,

- γ_R és la curvatura.
- x és la distància que separa el vehicle de la trajectòria.
- L és la longitud de corda de γ_R.
- θ és l'angle que forma respecte la trajectòria.

Com s'especifica a la secció 3.7, existeix una funció que retorna a partir d'una imatge, les distàncies $d1$, $d2$ i l'angle θ segons el criteri utilitzat.

$$[d1, d2, \theta] = Activitat53(Imatge)$$

Amb la següent expressió el robot mòbil modificarà la seva velocitat angular fins a arribar a obtenir una distància amb la línia horitzontal nula. Per tant, s'aconsegueix tenir el control de la trajectòria ja que el robot modificarà la seva curvatura de forma constant, avançant en línia recta.

$$\omega_r = V\gamma_R = V\frac{2}{L^2}[x\cos(\theta) - \sqrt{L^2 - x^2}\sin(\theta)]$$

Aquest mètode imposa restriccions per tal de garantir l'estabilitat del sistema,

$$det(sI - J) = 0$$

on,

$$J = \begin{bmatrix} 0 & -V & 0 \\ 0 & 0 & 1 \\ \dfrac{\varphi_x}{T} & \dfrac{\varphi_\theta}{T} & \dfrac{(\varphi_\gamma - 1)}{T} \end{bmatrix}$$

On les derivades parcials respecte $\omega_r = f(x, \theta, \gamma)$

$$\varphi_x = \left.\frac{\partial \omega_r}{\partial x}\right|_0 = \frac{2V}{L^2};\ \varphi_\theta = \left.\frac{\partial \omega_r}{\partial \theta}\right|_0 = \frac{-2V}{L};\ \varphi_\gamma = \left.\frac{\partial \omega_r}{\partial \gamma}\right|_0 = 0$$

Per tant,

$$J = \begin{bmatrix} 0 & -V & 0 \\ 0 & 0 & 1 \\ \frac{\varphi_x}{T} & \frac{\varphi_\theta}{T} & \frac{(\varphi_\gamma - 1)}{T} \end{bmatrix} = \begin{bmatrix} 0 & -V & 0 \\ 0 & 0 & 1 \\ \frac{2V}{TL^2} & \frac{-2V}{TL} & \frac{-1}{T} \end{bmatrix}$$

Si es desenvolupa l'expressió anterior,

$$det(sI - J) = det \begin{pmatrix} s & V & 0 \\ 0 & s & 1 \\ \frac{-2V}{TL^2} & \frac{2V}{TL} & s + \frac{1}{T} \end{pmatrix} = s(\frac{s}{T} + s^2 + \frac{2V}{TL}) + \frac{2V^2}{TL^2},$$

Finalment s'arriba a la conclusió que,

$$s^3 + \frac{1}{T}s^2 + \frac{2V}{TL}s + \frac{2V^2}{TL^2} = 0 \Rightarrow \frac{L}{TV} \geq 1$$

on

- T és el temps necessari per realitzar la captura i el tractament de la imatge.
- V és la velocitat en l'eix y.
- L és la longitud de corda de γ_R.

Com la velocitat en l'eix y és constant $V_y = 0.1\ m/s$, per garantir l'estabilitat del sistema, es fixa la longitud de la curvatura γ_R a $L = 40\ cm$.

4.2 Comportament seqüencial en l'àrea de treball

L'aplicació final d'aquesta recerca és aconseguir que el robot mòbil faci el seguiment d'una ruta determinada dins de l'espai de treball.

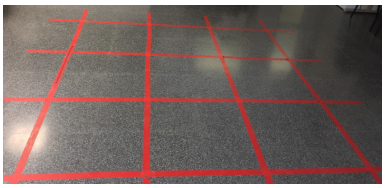

Figura 4.2.1: Espai de treball.

Es pot enfocar el problema com un comportament seqüencial del sistema. És a dir, com si es tractés d'una màquina d'estats on hi ha quatre possibles escenaris [4].

- S1: El robot avança indefinidament realitzant el seguiment de la línia, corregint les possibles desviacions d'angle i distància mentre no detecti una cruïlla.
- S2: El robot avança fins arribar a la cruïlla realitzant el seguiment de la línia, corregint les possibles desviacions d'angle i distància. Aquesta estapa es farà per odometria des del moment en que la imatge capturada registra una línia horitzontal.
- S3: El robot realitza un gir per odometria.
- S4: El robot s'atura ja sigui per la detecció d'un objecte, la finalització del trajecte o per ordre del usuari.

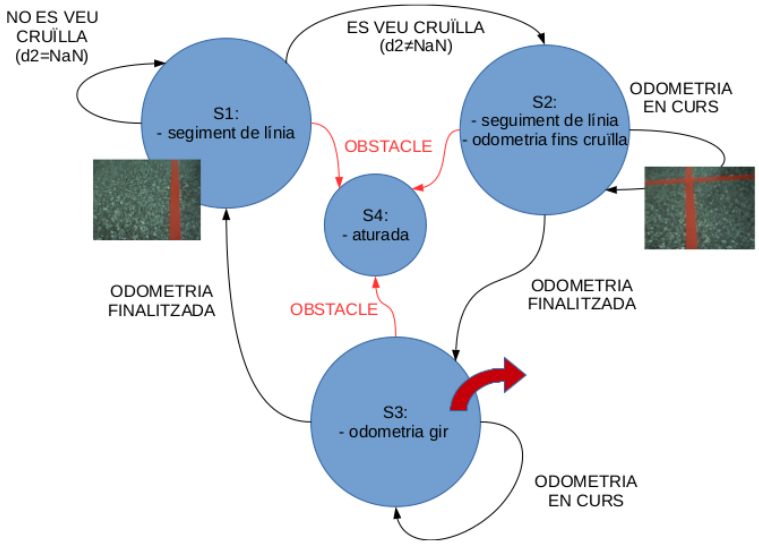

Figura 4.2.2: Màquina d'estats proposada.

Per problemes amb els paquets d'odometria del *Robotino* per *ROS* utilitzats a *MATLAB*, es substitueix utilitzar aquest mètode per el *Moviment Rectilini* en l'estat S2 i el *Moviment Circular* per l'estat S3.

4.2.1 Moviment Rectilini

El *Moviment Rectilini* és la trajectòria que descriu el moviment en una línia recta. En aquesta aplicació, es considera la velocitat del robot com constant, per tant, s'utilitza el *Moviment Rectilini Uniforme*.

$$x = x_0 + vt$$

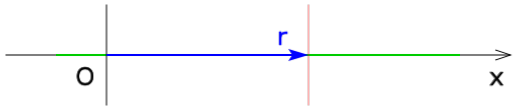

Figura 4.2.3: Moviment Rectilini.

En el moment en que a la imatge capturada es detecta una línia horitzontal, s'inicia el funcionament de l'estat S2 durant un temps limitat. Aquest temps, d'acord amb l'expressió anterior, és el temps que triga el robot en arribar a la posició desitjada. És a dir, on es situa la cruïlla.

4.2.2 Moviment Circular

El *Moviment Circular* és un desplaçament on el cos té una trajectòria circular al voltant d'un eix de gir. En aquesta aplicació, es considera la velocitat angular del robot com constant, per tant, s'utilitza el *Moviment Circular Uniforme*.

$$\varphi = \varphi_0 + \omega t$$

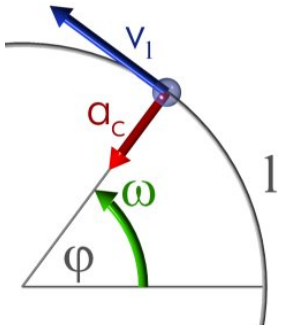

Figura 4.2.4: Moviment Circular.

En funció de la ruta a realitzar, en arribar a la cruïlla es possible que el gir a fer a S4 sigui de 0^o, -90^o, 90^o, o 180^o. Existeix una funció que té la finalitat de fer girar el robot durant un temps limitat segons el paràmetre d'entrada, *Sentit*, que és l'angle de gir.

$$Gir(Sentit)$$

Capítol 5

Planificació de Camins i Supervisió

5.1 Introducció

Aquesta secció és el punt més important de la recerca ja que integra tots els elements descrits fins aquí.
L'objectiu és tenir una aplicació en la qual l'usuari sigui capaç d'interactuar amb el robot. Aquesta eina ha de disposar de tals funcionalitats que l'usuari pugui emetre ordres al dispositiu i a l'hora supervisar el procés.

5.2 Planificació de camins

Com previament s'ha mencionat, el robot ha de ser capaç de realitzar una ruta deteminada dins de l'espai de treball. Donat un origen i un destí, existeixen varies solucions que compleixin les especificacions. L'objectiu del concepte *"Path Planning"* [5] és obtenir el trajecte més curt, eficient o amb el menor cost possible.
Aquesta tècnica té com a finalitat donar la solució més òptima dintre de les limitacions i característiques del sistema utilitzant algoritmes de cerca de 'grafs'.

Un graf, representa un conjunt de nodes units en una xarxa. Si dos nodes estan units, al viatjar d'un a l'altre es considera successor al node següent i predecessor al node que es deixa enrere. Normalment existirà un cost vincultat al desplaçament entre nodes. Un algoritme de cerca tractarà de trobar el camí més òptim entre dos nodes, com per exemple, un camí que minimitzi el cost de desplaçament o el número de passos a realitzar.
El cost vinculat al desplaçament entre nodes es guarda a la *matriu de distàncies*.

El tres mètodes de cerca de grafs utilitzats s'anomenen *Greedy*, *Dijkstra* i *A-Star*. A saber,

5.2.1 Greedy

La solució d'aquest mètode es va construint en etapes. A cada etapa s'afegeix un element nou a la solució parcial, el que es consideri el millor candidat en aquell moment. Les decisions triades no es poden revisar. És a dir, es selecciona l'element més prometedor sense pensar en futures conseqüències. Aquest mètode presenta més simplicitat i eficiència respecte els altres.

5.2.2 Dijkstra

La idea d'aquest algoritme consisteix en explorar tots els camins més curts que parteixen de l'origen i condueixen a la resta de nodes. Per cada iteració, partint de l'origen, busca el node que presenti un menor cost fins que arriba a l'objectiu. A cada pas, emmagatzema per una part el camí trobat i per una altre el pes del trajecte, trobant així la ruta més òptima.

5.2.3 A-Star

Aquest algoritme és més complet que els anteriors ja que considera els factors del cost total del trajecte i el valor heurístic dels nodes. Segueix la funció $f(n) = g(n) + h'(n)$ on $h'(n)$ representa el valor heurístic del node a evaluar, n, fins el final i $g(n)$ el cost real del camí recorregut per arribar des del node n al node inicial.

Considerant el següent exemple,

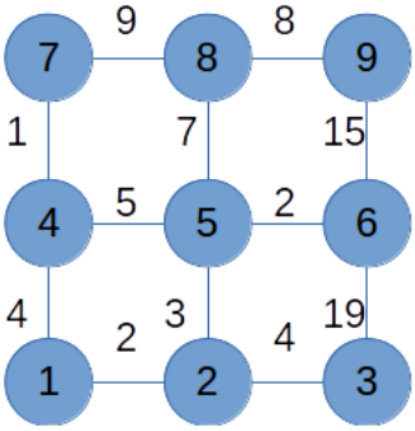

Figura 5.2.1: Exemple de "*Path Planning*".

La solució per a cada mètode és:

- *Greedy*: *ListOfWayPoints* =[1 2 5 6 9].
- *Dijkstra*: *ListOfWayPoints* =[1 2 5 8 9].
- *A-Star*: *ListOfWayPoints* =[1 2 5 8 9].

5.3 Supervisió

Una Interfície Gràfica d'Usuari(*GUI*) [6] és un programa informàtic que actua de interfície d'usuari. Utilitza un conjunt d'objectes gràfics per representar la informació i accions disponibles en la interfície. El seu principal ús consisteix en proporcionar un entorn visual senzill per permetre la comunicació amb el sistema operatiu d'una màquina o computador.

MATLAB GUIDE [36] és una eina per crear interfícies gràfiques d'usuari a l'entorn de *MATLAB* que permet la connexió amb altres mòduls del programa com per exemple el *Workspace*, *Simulink* o *SimEvents*.

En aquest cas, s'ha creat una interfície gràfica d'usuari per poder interactuar amb el *Robotino*. Aquesta interfície ha de tenir les funcionalitats necessaries per emetre ordes al robot i poder supervisar el procés en tot moment.

Es creu necessari que aquesta aplicació contingui els següents elements.

- Botons per començar/aturar el procés.
- Menú per seleccionar l'origen i el destí destijat.
- Menú de selecció del mètode per planificar la ruta.
- Botons per calcular/netejar la ruta seleccionada.
- Indicador de compliment o de necessitat de recalcular la ruta per obstacle detectat en el camí.
- Finestra amb la simulació del mapa de l'espai de treball.
- Botó per sortir de l'aplicació.

Per tal d'evitar l'aparició d'errors, s'aplica doncs, el concepte japonès del *"Poka-Yoke"* amb la finalitat d'imposibilitar l'error de l'usuari.

En primer lloc, a l'hora d'executar la funció *GuidePathPlanning_GuillermOsio.m*, apareix la següent finestra.

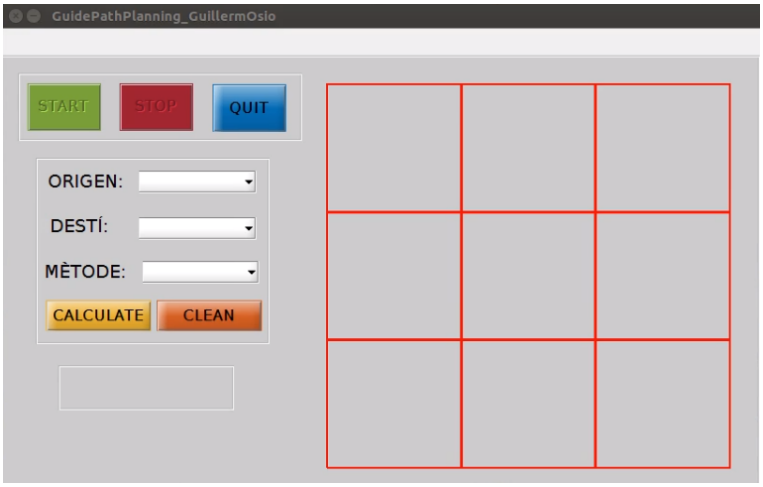

Figura 5.3.1: Interfície Gràfica d'Usuari en condicions inicials.

Es pot observar que els botons *Començar/Aturar* estan deshabilitats per evitar l'aparició d'erros d'execució. És imprescindible seleccionar un punt d'origen, un punt destí i un dels possibles mètodes (*Greedy*, *Dijkstra* o *A-Star*) per poder calcular la ruta.

A continuació, si es selecciona el botó "*Calcular*", en funció del mètode per fer la planificació, la ruta més òptima apareix reflectida a la finestra on es situa la simulació del mapa de l'espai de treball.

En aquesta situació es desitja anar del node 6 al 11 per el mètode *Dijkstra*.

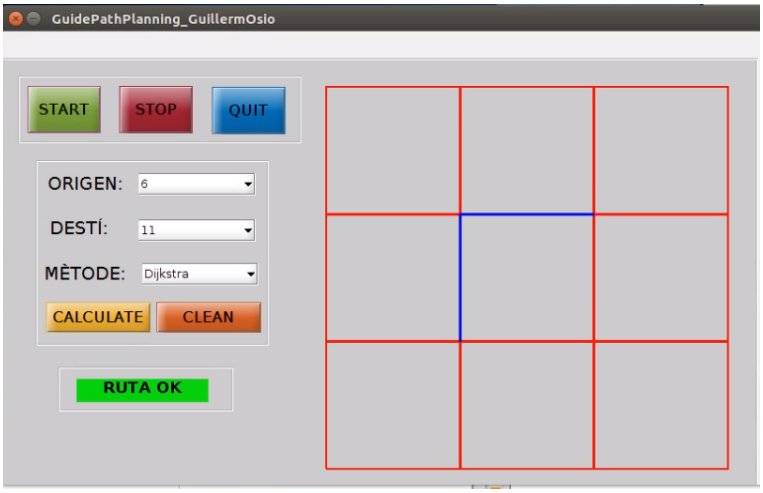

Figura 5.3.2: GUI amb una ruta disponible.

En aquest punt existeixen dos opcions:
a) Seleccionar el botó "*Netejar*", això permetria poder calcular una nova rutina, esborrant l'anterior.
b) Seleccionar el botó "*Començar*"per provocar que el robot mòbil comenci a realitzar la rutina destijada.

El robot sempre parteix de la posició *HOME* que equival al node 1. En primer lloc, anirà fins el node on inicia la rutina. A mesura que el robot avança, el tram que completa canvia de color per tal de saber la seva ubicació.

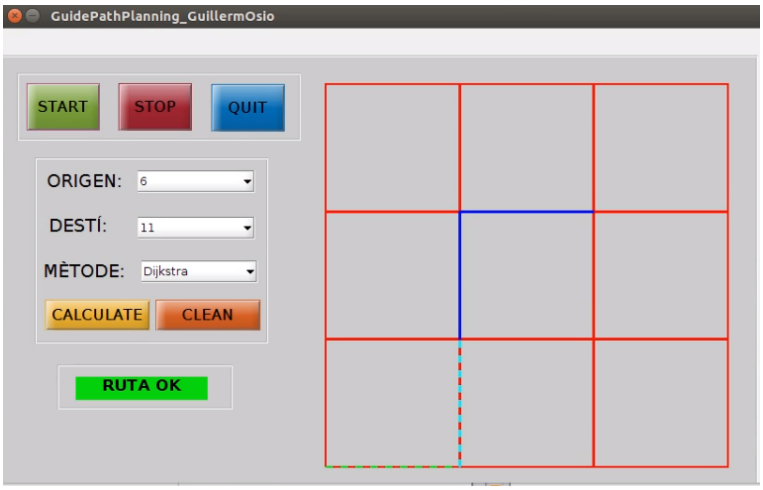

Figura 5.3.3: GUI amb ruta fins a inici.

En tot moment es pot aturar el procés amb el botó "*Aturar*" com a mesura de seguretat.

Una vegada el robot completa el trajecte inicial, procedeix a realitzar la ruta previament calculada.

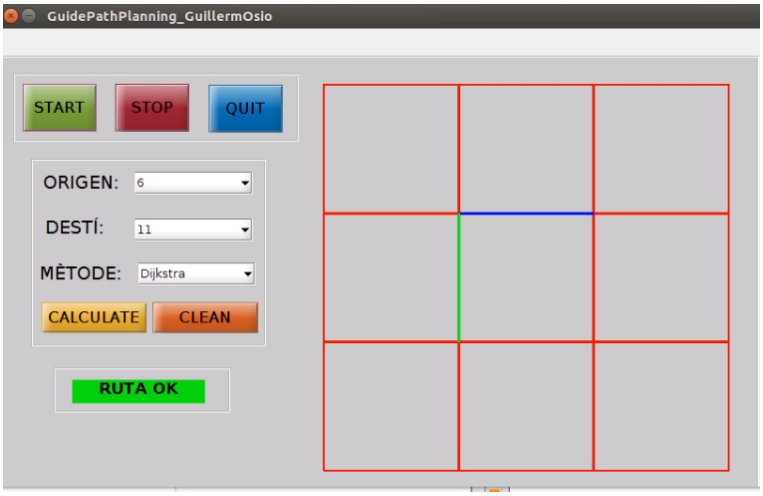

Figura 5.3.4: GUI amb una ruta en curs.

Tal i com s'ha explicat a la secció 4.2, el robot es pot aturar si detecta un objecte en el seu camí. Si aquest objecte no desapareix del trajecte durant un temps determinat, es procedirà a recalcular la ruta des del punt en el qual no ha pogut continuar amb la seva rutina fins a l'objectiu desitjat.
L'indicador canviarà el missatge "*RUTA OK*"a "*CANVI DE RUTA*". A més a més, la nova ruta apareixera reflectida al mapa.

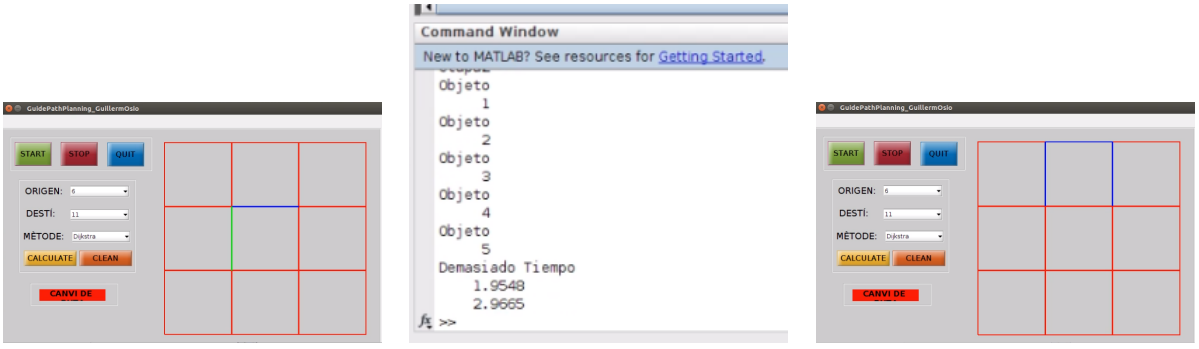

Figura 5.3.5: GUI recalculant ruta per obstacle.

Novament, continuarà amb la seva rutina fins a assolir l'objectiu.

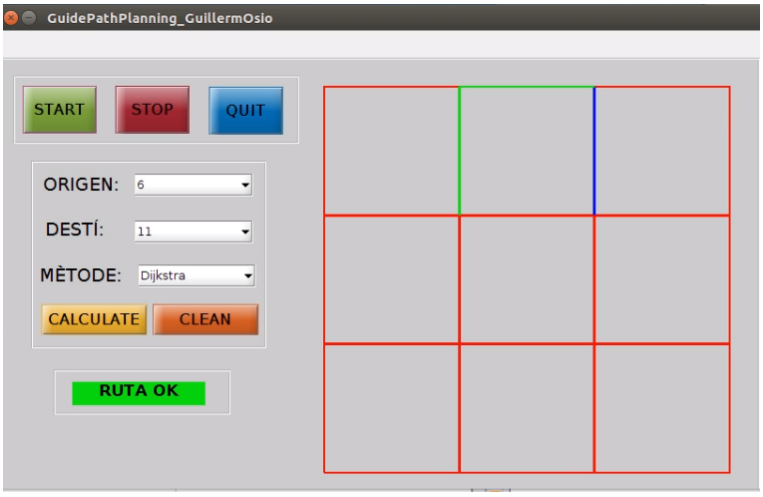

Figura 5.3.6: GUI amb una nova ruta en curs.

Finalment, el robot tornarà a la posició *HOME*.

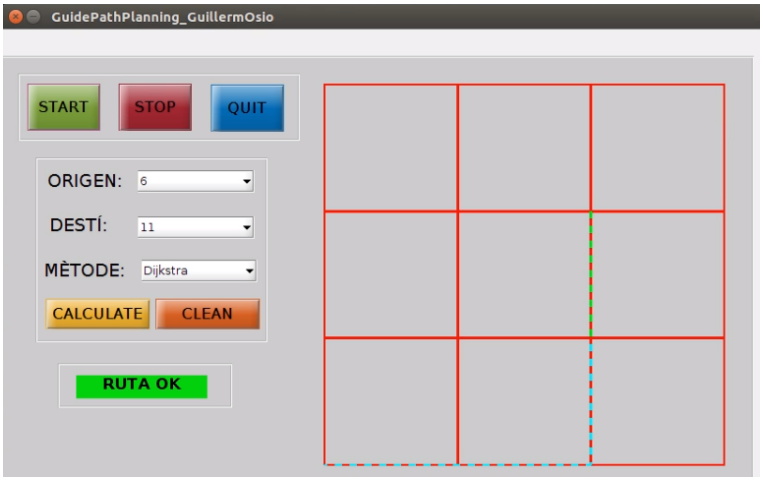

Figura 5.3.7: GUI tornant a *HOME*.

Capítol 6

Conclusions i Bibliografía

6.1 Conclusions

Les principals conclusions d'aquesta recerca són:

1. El procés d'instal·lació de la arquitectura del treball ha estat una feina difícil i que ha consumit la meitat del temps d'elaboració del TFG.
2. Caldría revisar a nivell intern l'estat de la xarxa ja que es produeixen interrupcions a l'hora de rebre informació dels sensors de proximitat que provoquen un mal funcionament del programa.
3. El canvi de plataforma sembla ser que ofereix millors resultats que amb el sistema anterior.
4. Els objectius plantejats per aquest TFG s'han assolit en la seva totalitat.

A nivell personal:

1. Les capacitats adquirides amb l'ús del software *MATLAB* s'han vist potenciades.
2. La elaboració de la memòria del TFG amb el "sistema"*LaTeX* ha representat un descrobriment a l'hora de redactar i organitzar el contingut d'un document.
3. S'ha adquirit un grau de maduresa degut a la afrontació d'una recerca individual.

6.1.1 Propostes de millora

Durant l'elaboració de la recerca s'han identificat alguns aspectes que caldría millorar. A saber:

1. Revisar a nivell de software el problema que genera utilitzar el paquet d'odometria per *ROS*. Seria millor treballar amb aquesta funció ja que pot presentar una major precisió.

6.2 Bibliografia

[1] Albert Masip-Àlvarez. Transformacions i rotacions d'eixos. Aplicació pràctica sobre el sistema de visió del Robotino Transformacions i rotacions d'eixos de coordenades necessaris per passar d'un píxel de la imatge capturada a un punt extern en coordenades robot. Novembre de 2015.

[2] Albert Masip-Àlvarez. Visió per computador. Apunts de Classe. Control i guiatge de robots mòbils. 2018

[3] Albert Masip-Àlvarez. Control de la trajectòria. Apunts de classe. Control i guiatge de robots mòbils. 2018

[4] Albert Masip-Àlvarez. PRÀCTICA 6: Control de trajectòria; seguiment de línia i acompliment de ruta. Apunts de classe. Control i guiatge de robots mòbils. 2018

[5] A. Masip-Alvarez. Finding the Minimum (cost) Path. Greedy, Dijkstra and A Path Finding Algorithms. Automatic Control Department. UPC. 2018.*

[6] A. Masip-A-Alvarez. Supervision. GUIDE and Simulink. Automatic Control Department. UPC. 2018.

[7] A. Ollero and G. Heredia. Stability analysis of mobile robots path tracking. International Conference on Intelligent Robots and Systems (IROS). Pp. 461 - 466. 1995. Pennsylvania - Pittsburgh, U.S.A.. Vol 3.

[8] Siegwart, Rolan; Nourbakhsh, Illah R. Introduction to autonomous mobile robots. Cambridge: MIT Press, 2004.

[9] RC. Gonzáles, R.E Woods, S.L. Eddings. Digital Image processinggusin Matlab. Pearson Prentice Hall. 2004

[10] R. Szeliski. Computer Vision: algorithms ans applications. Spinger, 2010 [En línea]. Disponible
http://szeliski.org/Book/

[11] D.H. Ballard, C.M. Brown. Computer vision. Prentice-Hall.1982

[12] J.Y Bouguet. Camera Calibration Toolbox for Matlab. 2015 [En línea].Disponible http://www.vision.caltech.edu/bouguetj/calib_doc / [Accés juny 2018].

[13] FESTO. Robotino. 2018. [En línea]. Disponible
http://www.festo-didactic.com/int-en/services/robotino/ [Accés abril 2018].

[14] Ubuntu. Ubuntu 16.04.4 LTS (Xenial Xerus). 2018 [En línea]. Disponible http://releases.ubuntu.com/16.04/ [Accés juny 2018].

[15] Ubuntu install of ROS Kinetic. 2017. [En línea]. Disponible http://wiki.ros.org/kinetic/Installation/Ubuntu Creative commons. [Accés juny 2018].

[16] ROS. Creating a workspace for catkin. 2017 [En línea]. Disponible http://wiki.ros.org/catkin/Tutorials/create_a_workspace [Accés juny 2018].

[17] ROSEnvironmentVariables. 2016 [En línea]. Disponible http://wiki.ros.org/ROS/EnvironmentVariables [Accés maig 2018].

[18] C. Verbeek. ROS-Robotino. 2016 [En línea]. Disponible http://wiki.openrobotino.org/index.php?title=ROS [Accés juny 2018].

[19] How to setup robotino for programming and navigation. 2016. [En línea].Disponible http://www.daslhub.org/unlv/wiki/doku.php?id=how_to_setup_robotino_for_programming_and_navigation [Accés juny 2018].

[20] Openrobotino.org Port 80. Index of /download/flashcard/. 2014. [En línea]. Disponible http://doc.openrobotino.org/download/flashcard/ [Accés juny 2018].

[21] C. Verbeek. Install daemons v2. 2013. [En línea]. Disponible http://wiki.openrobotino.org/index.php?title=Install_daemons_v2 [Accés juny 2018].

[22] C. Verbeek. Debrepository. 2016 [En línea]. Disponible http://wiki.openrobotino.org/index.php?title=Debrepository [Accés juny 2018].

[23] C. Verbeek. API2 source build. 2016 [En línea]. Disponible http://wiki.openrobotino.org/index.php?title=API2_source_build [Accés juny 2018].

[24] Robotino Node. 2016. [En línea].Disponible http://wiki.ros.org/robotino_node. Creative commons. [Accés juny 2018].

[25] MathWorks Aspectos destacados de la versión R2015b. 2018 [En línea]. Disponible https://es.mathworks.com/products/new_products/release2015b.html [Accés juny 2018].*

[26] The MathWorks, Inc. Robot Operating System (ROS) Support from Robotics System Toolbox. 2018 [En línea]. Disponible https://es.mathworks.com/hardware-support/robot-operating-system.html . [Accés juny 2018].

[27] MathWorks MATLAB. Imwrite. 2018. [En línea]. Disponible https://es.mathworks.com/help/matlab/ref/imwrite.html [Accés juny 2018].*

[28] MathWorks. MATLAB. Imread. 2018. [En línea]. Disponible

https://es.mathworks.com/help/matlab/ref/imread.html [Accés juny 2018].

[29] MathWorks. rgb2ycbcr. 2018 [En línea]. Disponible https://es.mathworks.com/help/images/ref/rgb2ycbcr.html [Accés maig 2018].

[30] MathWorks. imerode. 2018 [En línea]. Disponible https://es.mathworks.com/help/images/ref/imerode.html [Accés maig 2018].

[31] MathWorks. MATLAB. Strel. 2018. [En línea]. Disponible https://es.mathworks.com/help/images/ref/strel.html [Accés juny 2018].

[32] MathWorks. Edge. 2018 [En línea]. Disponible https://es.mathworks.com/help/images/ref/edge.html [Accés maig 2018].

[33] MathWorks. Hough. 2018 [En línea]. Disponible https://es.mathworks.com/help/images/ref/hough.html [Accés maig 2018].

[34] MathWorks. Houghpeaks. 2018 [En línea]. Disponible https://es.mathworks.com/help/images/ref/houghpeaks.html [Accés maig 2018].

[35] MathWorks. Houghlines. 2018 [En línea]. Disponible https://es.mathworks.com/help/images/ref/houghlines.html [Accés maig 2018].

[36] MathWorks Creación de apps con interfaces gráficas de usuario en MATLAB. 2018 [En línea]. Disponible https://es.mathworks.com/discovery/matlab-gui.html [Accés maig 2018].

Capítol 7

Annexos

En aquest apartat es pot trobar la guia d'instal·lació de l'arquitectura i les diferents funcions utilitzades per el control del robot.

7.1 Annex A. Instal·lació de l'arquitectura

7.1.1 Instal·lació del sistema operatiu ROS Kinetic

```
$sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.li
$sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523
$sudo apt-get update
$sudo apt-get install ros-kinetic-desktop-full
$sudo rosdep init
$rosdep update
$echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
$source ~/.bashrc
$sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

7.1.2 Creació espai de treball per a ROS

```
$sudo mkdir catkin_ws
$sudo chmod -R 777 catkin_ws/
$cd catkin_ws
$sudo mkdir src
$sudo chmod -R 777 src/
$catkin_make
$source devel/setup.bash
$echo $ROS_PACKAGE_PATH /home/guillermo/catkin_ws/src:/opt/ros/kinetic/share
$export ROS_MASTER_URI=http://mia:11311/
```

7.1.3 Instal·lació de API2 al Robotino

```
$su root
$Password: dorp6
$echo "deb http://doc.openrobotino.org/download/packages/i386 ./" > /etc/apt/sources.list
$echo "deb http://old-releases.ubuntu.com/ubuntu/ jaunty main restricted" >> /etc/apt/sources.list
$echo "deb-src http://old-releases.ubuntu.com/ubuntu/ jaunty main restricted" >> /etc/apt/sources.list
$apt-get update
$apt-get install rec-rpc-qt4.5.0 robotino-common robotino-daemons robotino-api2 robotino-examples
$Install these packages without verification [y/N]? y
$Do you want to install symlinks? -> Yes
```

7.1.4 Instal·lació de API2 a l'equip

```
$sudo apt-get install cmake cmake-curses-gui
$wget -qO - http://packages.openrobotino.org/keyFile | sudo apt-key add -
$sudo su
$echo "deb http://packages.openrobotino.org/xenial xenial main" > /etc/apt/sources.list.d/openrobotino.l
$sudo apt-get update
$sudo su
$echo "deb http://doc.openrobotino.org/download/packages/i386 ./" >

/etc/apt/sources.list
$echo "deb http://old-releases.ubuntu.com/ubuntu/ jaunty main restricted" >> /etc/apt/sources.list
$echo "deb-src http://old-releases.ubuntu.com/ubuntu/ jaunty main restricted" >> /etc/apt/sources.list
$sudo apt-get update
$cd catkin_ws
$mkdir source
$cd source
$mkdir rec_rpc api2 common daemons
$cd../build
$mkdir rec_rpc api2 common daemons
$cd ..
$svn co svn://svn.rec.de/openrobotino/api2/trunk source/api2
$svn co svn://svn.rec.de/servicerobotics/rec_rpc/trunk source/rec_rpc
$svn co svn://svn.rec.de/openrobotino/common/trunk source/common
$svn co svn://svn.rec.de/openrobotino/daemons/trunk source/daemons
$sudo apt-get install libqt4-dev libqt4-core libqt4-xml
1.
$cd build/api2
$ccmake ../../source/api2 (c e c e g)
$make install
$export ROBOTINOAPI2_64_DIR=/home/robotino/build/api2/install/usr/local/$robotino/api2/
2.
$cd ../rec_rpc/
$ccmake ../../source/rec_rpc (c e c e g)
$make install
$make CREATE_INSTALLER
$sudo dpkg -i rec-rpc_1.6.0_amd64.deb
3.
$cd ../common/
$sudo apt-get install robotino-api2 robotino-config robotino-common robotino-daemons
$ccmake ../../source/common (c e c e g)
$make install
$make CREATE_INSTALLER
$sudo dpkg -i robotino-common_1.1.11_amd64.deb
4.
$cd ../daemons/
$ccmake ../../source/daemons (c e c e g)
$make install
```

7.1.5 Instal·lació del paquet de drivers de ROS per Robotino a l'equip

```
$cd catkin_ws/src
$wstool init
$wstool set robotino_node --svn
http://svn.openrobotino.org/robotino-ros-pkg/trunk/catkin-pkg/robotino_node/
$wstool update
$rosdep install -r --from-paths ./src/
$Repetir per: robotino_description robotino_local_move robotino_msgs robotino_navigation
robotino_safety robotino_teleop
```

7.1.6 Finalització de la instal·lació

```
$ Descarregar carpeta rec: http://svn.openrobotino.org/api2/trunk/lib/rec/
$ Modificar tots els fitxers catkin_ws/src/robotino_node/include.
Canviar rutes per /home/guillermo/catkin_ws/rec/robotino/api2/nom_fitxer.h
$
Escriure en terminal
$catkin_make.
Apareixeran diferents errors. Es necessari corregir tots. Basicament es modificar els seguents arxius:
$catkin_ws/build/robotino_node/CMakeFiles/robotino_node.dir/build.make
$catkin_ws/build/robotino_node/CMakeFiles/robotino_node.dir/cmake_clean.make
$catkin_ws/build/robotino_node/CMakeFiles/robotino_node.dir/depend.internal
$catkin_ws/build/robotino_node/CMakeFiles/robotino_node.dir/DependInfo.cmake
$catkin_ws/build/robotino_node/CMakeFiles/robotino_odometry_node.dir/depend.internal
$catkin_ws/build/robotino_node/CMakeFiles/robotino_odometry_node.dir/CXX.include

$catkin_make
```

En el següent enllaç es troba el fitxer *Robotino_ROS* on es detalla més el procés d'instal·lació.

`https://drive.google.com/drive/folders/1TrtDTTo-bW7K0UJU5Q996Xk5uOiYfsfv?usp=sharing`

7.2 Annex B. Funcions utilitzades per el control

Les següents funcions han sigut programades amb *MATLAB*. L'ordre de dependència entre elles es decriu a la següent figura.

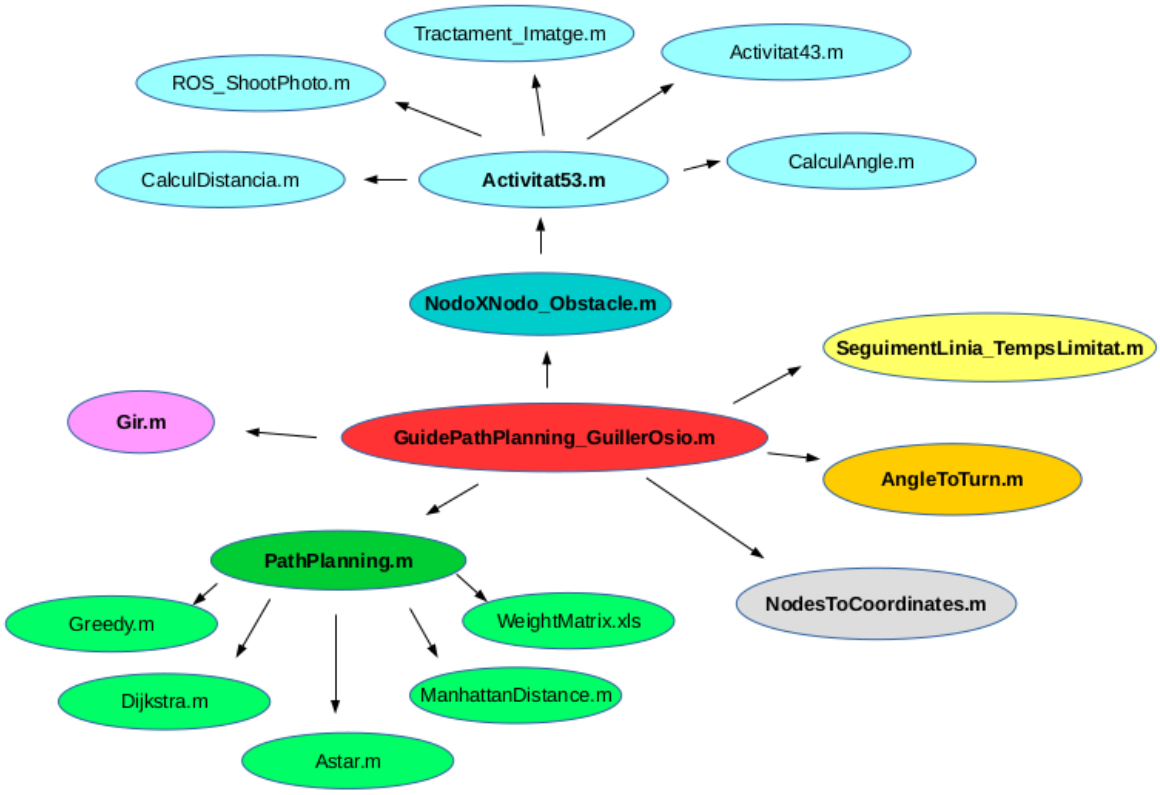

Figura 7.2.1: Dependència entre les diferents funcions.

Per raons d'extensió dels fitxers, s'adjunta el següent enllaç on es poden trobar aquestes funcions.

`https://drive.google.com/drive/folders/15os1jjKIvuMrllfK-Y-g5q_ww8ERssk6?usp=sharing`